US-PAT-NO: **6158019**

DOCUMENT-IDENTIFIER: US 6158019 A

TITLE: System and apparatus for merging a write event journal and an original storage to produce an updated storage using an event map

DATE-ISSUED: December 5, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Squibb; Mark D.	Republic	MO	N/A	N/A

APPL-NO: 08/ 971199

DATE FILED: November 14, 1997

PARENT-CASE:

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of the U.S. Provisional Patent Application No. 60/030,998, filed on Dec. 15, 1996, and entitled JOURNAL INDEXING SERVICES AND SERVICE KIT. The U.S. Provisional Patent Application No. 60/030,998 is incorporated herein in its entirety by reference thereto .

US-CL-CURRENT: 714/13, 714/20

ABSTRACT:

A method and apparatus for restoring an updated computer storage from a journal of write events and a copy of an original storage generates an event map from the journal of write events. The event map permits efficient combination of the contents of the write event journal and the original storage. The event map also enables translation of the event journal into a **delta** expressing the differences between the original and updated storages. The event map similarly permits efficient merging of a write event journal and an original file stored streaming tape.

51 Claims, 17 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 12

---------- KWIC ---------

Abstract Text - ABTX (1):

A method and apparatus for restoring an updated computer storage from a journal of write events and a copy of an original storage generates an event map from the journal of write events. The event map permits efficient combination of the contents of the write event journal and the original storage. The event map also enables translation of the event journal into a **delta** expressing the differences between the original and updated storages. The event map similarly permits efficient merging of a write event journal and

an original file stored streaming tape.

US Patent No. - PN (1):
6158019

Assignee Name - ASNM (1):
Delta-Tek Research, Inc.

Assignee Group - ASGP (1):
Delta-Tek Research, Inc. Montreal CA 03

Brief Summary Text - BSTX (6):
In a data processing system, a backup/restore subsystem, usually referred to as a backup subsystem, is typically used to save a recent copy of an active file and several earlier versions. There are, for example, three general strategies employed by backup systems. First, full backup periodically copies all files from a client system's storage to a backup server. A second strategy includes incremental backup, where the client system copies only the modified files to the backup server. In a third strategy, a **delta** backup copies only the modified portions of the modified files to the backup server.

Brief Summary Text - BSTX (7):
Complete discussions of various backup and storage technologies are disclosed, for example in U.S. Pat. No. 5,479,654 entitled APPARATUS AND METHOD FOR RECONSTRUCTING A FILE FROM A DIFFERENCE SIGNATURE AND AN ORIGINAL FILE, which is hereby incorporated by reference. Applicants co-pending applications entitled COMPUTER APPARATUS AND METHOD FOR MERGING A SEQUENTIAL PLURALITY OF **DELTA** STREAMS, Attorney Docket No. HP-10951196-1, filed Nov. 14, 1997, and COMPUTER APPARATUS AND METHOD FOR MERGING SYSTEM DELTAS, Attorney Docket No. HP-10960109, filed Nov. 30, 1995, also relate to backup and storage technologies and are hereby incorporated by reference. In addition, the Storage Service Kit, .COPYRGT.1996 by Mark Squibb also relates to data storage systems and is hereby incorporated by reference.

Brief Summary Text - BSTX (8):
It is apparent to those skilled in the art that in any given backup system, the higher the backup frequency, the more accurate the backup copy will represent the present state of the data within a file. Considering the large volume of data maintained and continuously generated in a data processing system, the amount of storage, **time** and other resources associated with protecting data are very substantial. Thus, those skilled in the art are continuously engaged in searching for better and more efficient ways to provide data protection.

Brief Summary Text - BSTX (9):
The **time** lag between the last backup, for example by the backup methods described above, and the current data on an active system represents risk of data loss. This "protection gap" is an active concern among computer users because it represents unprotected information. Mirroring systems, described below, partially overcome this gap.

Brief Summary Text - BSTX (10):
It is well known in the art to capture write events to a storage system. For example, each **time a change** is made to a storage device, the **change** is recorded or logged into a second media. A variety of types of media have been used for recording of logs including, for example, streaming tape, hard disk, and remote hard disk.

Brief Summary Text - BSTX (11):
A write event comprises, for example, a storage indicator indicating what

for the journal of Malcolm to be used to **recover** a file by the specified method: the base file must first be placed on random access media; a seek to byte 1000 must occur followed by a write of the 100 bytes from the first write event; and the primary media must than seek to byte 500 and the 100 bytes relating to the second write event must be written to the primary media. This requirement to seek in the base file multiple times effectively prevents streaming media from being used in combination with Malcolm's journals or in the common precedent of using replaying database redo logs.

Brief Summary Text - BSTX (26):
In the field of information storage, a variety of media types are used. Streaming media, or tape, is dramatically cheaper than random access media. For most practical purposes, however, tape is not considered a **readily** seekable media. While most tape devices support positioning of media to a linear address, this positioning requires linear traversal of a very long media. This positioning takes a lot of **time**, and is used sparingly in practical applications.

Brief Summary Text - BSTX (35):
It is a further object of the present invention to provide a cost and **time**-effective method for providing an archive mirror using inexpensive streaming media.

Brief Summary Text - BSTX (36):
It is a further object of the present invention to convert a write event journal into a **delta**.

Brief Summary Text - BSTX (38):
It is a further object of the present invention to enable use of a **read**-only base stream and an event log as a readable, seekable and writeable stream.

Brief Summary Text - BSTX (39):
It is a further object of the present invention to provide an apparatus and method for combining a plurality of write event journals with a **read**-only non-seekable base stream to produce an updated stream.

Drawing Description Text - DRTX (12):
FIG. 4A illustrates exemplary components for fulfilling a **read** request according to an embodiment of the present invention.

Drawing Description Text - DRTX (13):
FIG. 4B illustrates an exemplary method for fulfilling a **read** request according to an embodiment of the present invention.

Drawing Description Text - DRTX (15):
FIG. 5 illustrates an exemplary method for converting an event journal to a **delta** according to an embodiment of the present invention.

Drawing Description Text - DRTX (16):
FIG. 6A illustrates an exemplary flow chart showing how to use a **read**-only storage and an event journal as a seekable, readable and writeable storage according to an embodiment of the present invention.

Drawing Description Text - DRTX (17):
FIG. 6B illustrates an exemplary flow chart showing how to write to a **read**-only storage and event journal combination according to an embodiment of the present invention.

Drawing Description Text - DRTX (18):
FIG. 6C illustrates a **read from a read**-only storage and event journal

combination according to an embodiment of the present invention.

Detailed Description Text - DETX (5):

FIG. 1C illustrates an original computer storage 1 containing original data 6. A collection of write events, for example indicated at 8, 9, 10, 11, 12 each result in a **change** to computer storage 1 by overlaying data already there. For example, each write event including the data written and an address from the origin of the computer storage is indicated by 8-14, 9-17, 10-16, 11-18, 12-15 as shown in FIG. 1C. The series of write events 13 resulting in a **change** in the data of the original storage 6 results in an updated storage 19.

Detailed Description Text - DETX (8):

For the purposes of simple illustration, all of the write events in this description apply to a single computer storage. The same methods also apply, however, to a computer or network of computers each having a plurality of storages. For systems having a plurality of storages, it is common to include checkpoint events. Checkpoint events contain markers that indicate stable or committed instants where data in a plurality of storages is synchronized or valid. Checkpoints often facilitate **recovery** to a particular point in **time** for systems having a plurality of storages.

Detailed Description Text - DETX (21):

Event markers are stored in order of event marker address. Sorting enables rapid location of markers relating to an event marker address. The process of inserting and deleting whole markers in the list can be **time** consuming, especially if the list or btree is large. The present invention practices two techniques for improved performance when lists become large. The first practice is known as marker editing. Editing modifies an existing entry in a list when it is known that the edits do not affect the sequence represented by the list. In most cases, editing an existing marker is many times faster than deleting and reinserting a tree entry or sorted list entry.

Detailed Description Text - DETX (27):

The event map according to the present invention is useful for a variety of purposes. It is well known to create a backup of a computer by copying an original storage to a streaming media. Prior art systems describe methods for **recovering** from an event journal by copying a backup onto a hard disk and "replaying" the events in an event journal. This technique only works, however, if the number of events in the event journal is small enough to replay in a reasonable amount of **time**. For example, if a large volume of changes are stored in an event journal, replaying the entire event journal to recreate a file could take a prohibitively long **time**. Thus, such a technique is impractical for sustained off-site backup maintenance. The requirement to periodically refresh the entire backup creates a huge amount of network traffic and disqualifies this method from use for large systems. In addition, the prior art systems require the intermediate step of placing the original data file on a seekable medium prior to replaying the event journal to recreate a file. On conventional tape back-up systems, however, only a small amount of disk space is available, if at all, and thus the original file cannot be placed on disk for merging with an event journal as is done via an event map according to the present invention.

Detailed Description Text - DETX (28):

In contrast, the event map of the present invention enables efficient updating of a backup stored on streaming media. For example, by creating the event map according to the present invention, the **net** result of the changes in the event journal are combined with the original file, thus reducing the amount of network traffic associated with the back-up or recreation process and there is no requirement for an intermediate step of placing the original file on a seekable medium as the event map can be combined sequentially with the original

file.

Detailed Description Text - DETX (29):

As indicated earlier, it is well known in the art that a backup comprises a copy of an original storage and that backup copies are often stored on streaming media because streaming media is cheaper than random access media. It is also well known to store an event journal. For example, in a conventional computer system with a primary computer and a backup computer connected to a network, a copy of a base (e.g., original) file is copied from the primary computer to the backup computer. To generate a backup copy of the current state of the file, the base file would be written to a disk from the backup computer for combination with the changes to the base file, stored as an event journal on the backup computer, thus necessitating many I/O operations as described earlier. The updated file would then be stored in the backup system. In addition to requiring transfer from a backup streaming media to a disk to generate an updated backup copy of a file, such a backup system also generally does not provide the capability to incorporate only recent changes to the base file, which may no longer exist on the backup system if replaced following a backup operation. Thus, it is not known to merge data in an event journal with an original storage stored on streaming media for **recovery**. It is further not known to merge data from an event journal with data in a streaming media for maintenance of a backup copy to keep the backup copy up to date in accordance with the present invention.

Detailed Description Text - DETX (32):

FIG. 4A illustrates exemplary components of the present invention that participate in fulfilling a **read** request 30 for an updated storage, an original storage 6, an event journal 21 and an event map 29. The flowchart of FIG. 4B describes an exemplary method for fulfilling a **read** request from the combination of FIG. 4A comprising an original storage 6, an event journal 21 and an event map 29. A **read** request is composed, for example, of two elements: a data position; and a **read** size. The data position gives, for example, the starting address relative to an origin of the data to be **read**. The **read** size gives, for example, the count of elemental units to be obtained from the storage. The sum of the data position and the **read** size gives the address of the ending **read** address.

Detailed Description Text - DETX (33):

With reference to FIG. 4B, the first step 66 in processing a **read** request is to determine the data position, **read** size, and ending **read** address 32. The event map is queried for a marker that contains the current **read** position in step 67. If no marker references the current **read** position in step 68, the number of storage units until the next **read** marker is retrieved in step 72, the next marker count. The unit **read** size is calculated to be the minimum of the next marker count and the **read** size in step 73. Data from the original storage is copied into the **read** buffer to fulfill the unit **read** count of primary storage elements in step 74.

Detailed Description Text - DETX (34):

If an event marker is found which corresponds to the **read** position in step 68, the unit **read** size is calculated to be the minimum of the overlapping marker segment size and the **read** size in step 69. The marker data pointer is used to locate the corresponding event data in the event journal and fulfill the request for unit **read** size from the event journal in step 70. Next, the **read** size is decremented by the number of elemental storage units fulfilled, unit **read** size, in the last iteration, and the **read** position is advanced by the unit **read** size to indicate partial fulfillment of the **read** request in step 71. When the **read** size reaches zero, the **read** is fulfilled in step 75 and the process terminates in step 76. If the **read** size is not zero, the process resumes by querying the event map in step 67.

Detailed Description Text - DETX (35):

Application of the **read** method according to the present invention to cause sequential **reading** of an updated stream from beginning to end is an efficient way to merge an original stream and an event journal. FIG. 4C shows an exemplary flow chart that generally describes the method according to the present invention. This method is further disclosed in source code form at, for example, pages 9-10, 15-18, 19-20, 23-26, 30-31 and 33-34 in the attached paper appendix. The source code references four similar but distinct uses of the present invention. Each of these several behaviors can be, for example, invoked by program options.

Detailed Description Text - DETX (36):

As illustrated in FIG. 4C, in a method for merging a non-seekable base stream with an event log, an event map is constructed from the event journal as described above in step 78. A copy of an original storage on a streaming media is loaded into a tape drive in step 79. A series of **read** requests requesting consecutive segments of data from the updated storage represented by the combination of the original storage and the event journal are issued and fulfilled by, for example, the method of FIG. 4B in step 80. The results of the **read** requests are subsequently recorded to a target storage which may be another streaming media, disk or other storage in step 81. The process continues until complete in steps 82 and 83.

Detailed Description Text - DETX (37):

The sequential **read** process, the subject of FIG. 4B and source code disclosure at, for example, page 20 the attached paper appendix, causes the copy of the original storage to be consumed from beginning to end. The seeks which occur to the original storage advance the original storage beyond the same number of primary storage units supplied by the event journal. As a result, the original storage is consumed from beginning to end without seeking notwithstanding skipping of data units provided by the event journal. This characteristic enables efficient combination of an original computer storage and an event journal.

Detailed Description Text - DETX (38):

Seeks on the original storage serve to skip data segments provided by the event journal. Co-pending application by applicant, entitled COMPUTER APPARATUS AND METHOD FOR MERGING A SEQUENTIAL PLURALITY OF **DELTA** STREAMS recites a method and apparatus to capture skipped segments into an inverse **delta**. When processing a stream from beginning to end, the act of discarding characters is awkward. In practice, seeks in the original media only occur when a segment from the original storage has been overwritten. The normal effect of this on an original stream is to skip the overtyped characters. As recited above, the means of skipping overtyped characters is to discard them. The present invention includes, for example, methods compatible with co-pending application entitled COMPUTER APPARATUS AND METHOD FOR MERGING A SEQUENTIAL PLURALITY OF **DELTA** STREAMS which, for example, captures an "inverse **delta**" which is a list of changes that if made to the updated storage convert it back into an original storage. The present invention also produces inverse deltas. The method simply requires capturing elemental storage units skipped in the original stream as mismatch segments and recording data segments used from the original stream as matching segments.

Detailed Description Text - DETX (39):

It is similarly an object of the present invention to translate an event journal into a **delta**. A **delta** contains, for example, alternating frames describing matching and mismatching sections of an original and updated storage. The method is disclosed in source code with reference to, for example, pages 12, 21-22, 32, and 33-34 of the attached paper appendix with

particular reference to the class named JournalDelta.

Detailed Description Text - DETX (40):
The flow chart of FIG. 5 illustrates an exemplary embodiment of a method for converting an event journal to a **delta** according to the present invention. An event map is constructed for the event journal in step 85. A variable tracking the logical progress through the updated stream is initialized in step 87. This variable tracks the position accounting of the updated file. This logical position advances resulting from an accounting for storage units in the updated stream. Each **time** this position advances, the curposition variable is advanced in step 94.

Detailed Description Text - DETX (43):
Finally, the generated frame is recorded in step 94 and the process resumes. The curposition is incremented to account for data represented by the current frame in step 94 and the process resumes by checking if the storage is complete in step 88. If not, the process above repeats until all elemental storage units of the updated storage are accounted for. The **delta** of the present method is particularly useful when used in conjunction with co-pending applications entitled COMPUTER APPARATUS AND METHOD FOR MERGING A SEQUENTIAL PLURALITY OF **DELTA** STREAMS and COMPUTER APPARATUS AND METHOD FOR MERGING SYSTEM DELTAS.

Detailed Description Text - DETX (44):
Read-only files are well known in the art. They are common to write-once media such as CD-ROMs and the like as well as network file systems where a user may lack permission to or the ability to modify a particular storage. The present invention further provides a means of using a combination of a **read**-only storage, an active event journal and an event map as a seekable-readable-writeable storage. The method is generally disclosed in source code at, for example, pages 9-10, 15-18, 19-20, 23-26, 30-31 and 33-34 of the attached paper appendix. The flowcharts of FIG. 6A-6C generally describe an exemplary method according to the present invention.

Detailed Description Text - DETX (45):
The method of FIG. 6A includes, for example, the step of initializing an event journal in step 97. Initialization may be, for example, creation of a new event journal or activation of an existing journal. If the session refers to a continuation of an earlier session, the storage associated with the event journal is opened for **reading** and writing. If this is a new session, an event map is created, otherwise an earlier event map is activated in step 98. The event map and the event journal should be consistent. Note that if an event journal exists but no event map exists, the above method for generating an event map from an event journal is used. The final step is to open the **read** only storage in step 99. Note that by definition the **read**-only cannot be modified.

Detailed Description Text - DETX (46):
After initialization of the event journal, event map and **read** only storage, **read** and write accesses to the storage are performed as generally described in step 100, and specifically performed as described in FIGS. 6B and 6C. With further reference to FIG. 6B, the present invention diverts writes that would normally apply to the **read**-only storage to the event journal. This diversion is performed by first constructing a write event entry from the write request by determining the data and position represented by the write request. The data position is used as the event address. The data included in the write request is used as the event data.

Detailed Description Text - DETX (48):
Read requests to the combined **read**-only storage, event journal combination

are generally processed using the method illustrated, for example, in the flowchart of FIG. 6C. Instead of **reading** the source file, the **read** request is diverted and fulfilled by the method generally represented by FIG. 4B. The combination of using this **read** and write method provides a readable and writeable interface to a **read**-only storage.

Detailed Description Text - DETX (49):
The technique according to the present invention can be used, for example, to provide a plurality of interfaces to a **read**-only file. Consider, for example, a group of users all having access to a **read**-only storage but desiring to make changes to this storage. The method according to the present invention can be applied for each user who generates an independent event log that contains only the changes made by the user. These changes are invisible to the other users permitting each user to **change** his data view as necessary.

Detailed Description Text - DETX (50):
A similar application of the present invention uses the above method for simulation of a standard file interface using only a **read** only original storage and an event log in the absence of the **read** only file. If the **read**-only file above contains no data then the event journal contains all of the subject data. This capability permits a readable writeable and seekable file system to be created on a seekable write-once media like a CD-ROM. The method involves creating the event journal on CD-ROM and using the **read** and write simulation methods disclosed in the previous section to fulfill all **read** and write requests.

Claims Text - CLTX (2):
reading each of a plurality of write event entries from an event journal;

Claims Text - CLTX (13):
10. The method according to claim 1, wherein the steps of generating the current event marker and determining if an overlap condition exists are performed immediately after the step of **reading** each of the plurality of write event entries.

Claims Text - CLTX (15):
12. The method according to claim 1, further comprising the step of dividing the event journal into a plurality of segments and wherein the step of **reading** each of the plurality of write events includes **reading** each of the plurality of write events for a respective one of the plurality of segments.

Claims Text - CLTX (16):
13. A method of fulfilling a **read** request for an updated storage using an original storage, an event journal and an event map, the method comprising the steps of:

Claims Text - CLTX (17):
receiving a **read** request, the **read** request including a data position and a **read** size;

Claims Text - CLTX (18):
identifying, from the **read** request and via the event map, portions of the **read** request to be provided by the event journal and portions of the **read** request to be provided by the original storage; and

Claims Text - CLTX (19):
fulfilling the **read** request.

Claims Text - CLTX (20):
14. The method according to claim 13, wherein the data position represents

an offset from an origin of the updated storage and the **read** size represents a number of storage units to be **read** from the updated storage, the updated storage containing a data content of the original storage and subsequent changes to the original storage via a plurality of write events.

Claims Text - CLTX (24):
reading each of the plurality of write event entries from the event journal;

Claims Text - CLTX (29):
20. The method according to claim 13, wherein the original storage includes a **read** only storage device.

Claims Text - CLTX (30):
21. The method according to claim 13, wherein the step of fulfilling the **read** request includes **reading** the requested data from a respective one of the original storage and the event map as determined via the identifying step.

Claims Text - CLTX (31):
22. The method according to claim 21, wherein the step of fulfilling the **read** request further includes **reading** each contiguous segment of the original storage and recording a result of the **read** request on a second storage media.

Claims Text - CLTX (33):
24. The method according to claim 13, wherein the original storage resides on a streaming tape and the streaming tape is not repositioned during the step of fulfilling the **read** request except to skip overlaid data segments from the event journal.

Claims Text - CLTX (35):
26. A method for converting an event journal into a **delta**, comprising the steps of:

Claims Text - CLTX (52):
38. The method according to claim 1, wherein the event journal includes a plurality of event journals and the steps of **reading**, generating and determining are performed for each of the plurality of event journals.